Combining parallel techniques for Cloud-Based SMT Solving

Matteo Marescotti*

Università della Svizzera Italiana, Switzerland

Abstract

Satisfiability Modulo Theories (SMT) is a highly expressive language that combines propositional satisfiability with first-order logics allowing the modelling of constraint problems arising from practical domains in a natural way. The increasing popularity of SMT requires SMT solvers to be capable of handling increasingly complex constraint problems. This paper shows how cloud computing could speedup SMT solving by combining together different parallel techniques successfully used in constraint solving. In particular, the experiments performed on OpenSMT2 show that a proper parallel configuration could result, on average, in twofold speedup.

1 Introduction

An essential tool in formal verification is the constraint solver that is used to prove the existence or not of a combination that satisfies the provided constraints. The *SAT solver* is the basis of a constraint solver, it is used to prove or disprove the properties of the system being modeled. State-of-the-art SAT solvers implement the *Conflict-Driven Clause Learning* (CDCL) [6] algorithm which extends the original *Davis–Putnam–Logemann–Loveland* (DPLL) [2] procedure with conflict analysis [9].

Constraints obtained from source code during verification processes almost always contain a combination of different theories such functions, arithmetics, arrays, etc. therefore the need of SMT (Satisfiability Modulo Theories) to natively extend SAT with such theories becomes essential.

The SAT problem is proved NP-complete and the introduction of background theories can only make the process even harder. However, there exists relatively little research on parallel SMT solving, leaving could computing environment and parallel CPUs architectures unexploited. Moreover SMT is a very expressive formalism applicable to software verification as well as program synthesis, program repair, and constraint programming. Thus, improving SMT solving will naturally improve all the applications in which it is found useful.

There are two approaches used to exploit parallel computing in constraint solving: *portfolio* [3] and *search-space partitioning* [4]. Both approaches are widely used for SAT solving, but there exists relatively little research on how good they perform applied to SMT solving. In my PhD I study how these parallel approaches combined can be beneficial targeting specifically SMT solving; the next sections show some details about my current research status.

2 Related Work

Regarding SMT, a pure portfolio approach is used by the SMT solver CVC4 [1] and designed to run in a single machine. A more complex portfolio combined with clause sharing has been implemented using the SMT solver Z3 [8]. A divide-and-conquer approach has been implemented

*Supported by the Swiss National Science Foundation (SNSF) grant 153402.

on top of the SMT solver Boolector [7]. Despite the good results, all these works only target a single parallel approach and are limited to execution on single machines, preventing the system to scale to more challenging problems.

3 Parallel SMT Solving

Both SAT and SMT solvers use heuristics to guide the search over the boolean space, but the intrinsic difficulty of the problem makes every possible heuristic overall inaccurate, therefore small changes in the heuristic can result in significant differences in run times.

The portfolio approach draws its advantages from this random behaviour: several solvers are executed concurrently and each one with a different initial seed to increase the chance that one of them finds a solution in short time by covering as much search-space as possible quickly. However this property doesn't belong to all the instances, for example many instances require a minimum number of decisions under which a solution is never provided. Therefore a concurrent execution is almost useless. A big improvement to portfolio is to share the learned clauses between the solvers, this with the aim to prevent solvers to cover search-space previously covered by other solvers unsuccessfully.

Search-space partitioning partitions the input formula into a fixed number of partitions in such a way that they do not share any model among them and whose disjunction is equisatisfiable. The partitions can be solved independently using parallel computing. Again the result relies on the heuristic used for partitioning that sometimes produces more difficult partitions than the original formula.

3.1 Parallelization Framework

Neither Portfolio nor search-space partitioning alone is always the best choice, there is the need of combining them in order to complement their deficiencies.

Fig. 1 gives an overview of the parallelization framework based in our earlier work [5] extended with clause sharing feature. The architecture follows a client-server structure: the server receives input formulas encoded in the SMTLIB2 format[1] from the user and handles the connection with the clients. Each client consists of a SMT solver wrapped by a network layer. Once the formulas are received, the server according to its configuration partitions them in one or more partitions using the *partition heuristic* and each client is then provided with a partition. With a proper configuration the framework achieves to combine portfolio and search-space partitioning by providing each partition to many solvers. Finally clause sharing is also supported by the framework with the aid of the *clause database* containing the clauses sent by the solvers for a given SMT instance, and the filter and selection heuristics that respectively filters the clauses published by the solvers and selects the most valuable to update the solvers.

4 Experiments

The framework is thought to be as modular as possible in order to make the SMT solver and the heuristics easily replaceable. In the current implementation the only supported solver is OpenSMT2, which limits the framework to support only quantifier-free theories of uninterpreted function with equalities (QF_UF) and linear real arithmetics (QF_LRA). Nevertheless, to the best of my knowledge this is still the most versatile parallel implementation capable of handling

[1] http://smtlib.cs.uiowa.edu

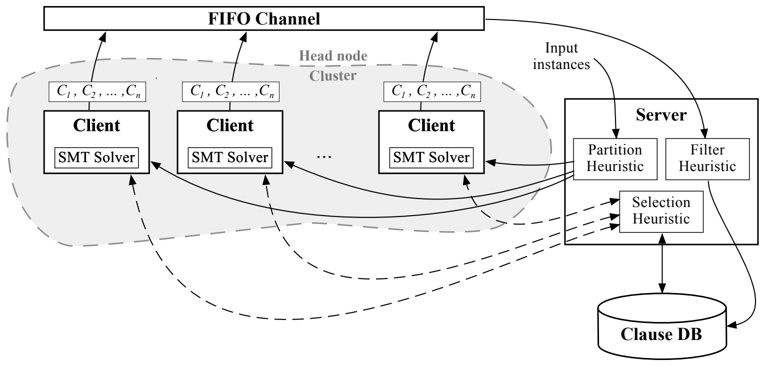

Figure 1: The parallel SMT solver framework with clause exchange mechanism.

two theories. The filter heuristic stores only clauses with less than a fixed number of literals: in our experiments the threshold is set to five. The selection heuristic is random. Moreover the server handles safely new client connection and client disconnection.

The experiments were performed in a cluster with eight compute nodes, each one equipped with two CPU Quad-Core AMD Opteron 2384 and 16GB of RAM. Each node runs eight clients for a total of 64 solvers connected to the server. The timeout is fixed to 1000 seconds wall time.

The results show that clause sharing improves performance better with pure portfolio: due to the high number of solvers and different taken paths, the clause set has an higher quality. The speedup due to clause sharing is 2.05 times, solving 10 more instances within the timeout (Fig. 2).

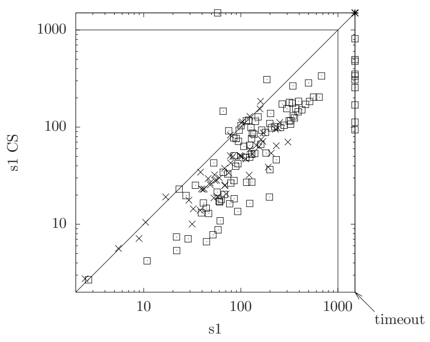

Figure 2: Portfolio with 64 solvers (s1) with against without clause sharing (CS) using filtering heuristic threshold 5.

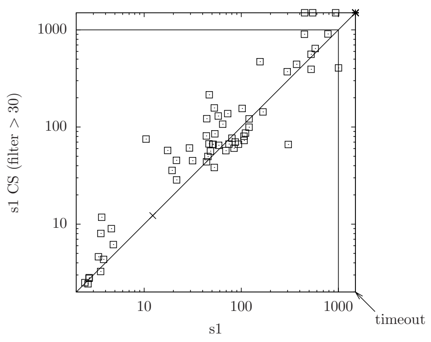

Figure 3: Portfolio with 64 solvers (s1) with against without clause sharing (CS) using filtering heuristic threshold 30.

5 Future Work

We experimented different filtering heuristics: Fig. 3 shows that a loose filtering performs worse than simple portfolio. This result shows how crucial is the heuristic choice also regarding clause sharing, and suggests an interesting and novel research direction on how to improve such heuristics.

References

[1] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli. CVC4. In *Proc. CAV 2011*, volume 6806 of *LNCS*, pages 171 – 177. Springer, 2011.

[2] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[3] Y. Hamadi, S. Jabbour, and L. Sais. ManySAT: a parallel SAT solver. *Journal on Satisfiability Boolean Modeling and Computation*, 6(4):245 – 262, 2009.

[4] A. E. J. Hyvärinen, T. A. Junttila, and I. Niemelä. Partitioning SAT instances for distributed solving. In *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings*, pages 372–386, 2010.

[5] A. E. J. Hyvärinen, M. Marescotti, and N. Sharygina. Search-space partitioning for parallelizing SMT solvers. In *Theory and Applications of Satisfiability Testing, SAT 2015, 18th International Conference, Austin, TX, USA, September 24-27, 2015. Proceedings*, pages 369–386, 2015.

[6] J. P. Marques-Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[7] C. Reisenberger. PBoolector: a parallel SMT solver for QF_BV by combining bit-blasting with look-ahead. Master's thesis, Johannes Kepler Univesität Linz, Linz, Austria, 2014.

[8] C. M. Wintersteiger, Y. Hamadi, and L. M. de Moura. A concurrent portfolio approach to SMT solving. In *Proc. CAV 2009*, volume 5643 of *LNCS*, pages 715–720. Springer, 2009.

[9] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient conflict driven learning in boolean satisfiability solver. In *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2001, San Jose, CA, USA, November 4-8, 2001*, pages 279–285, 2001.